24-678: Computer Vision for Engineers

Carnegie Mellon University

PS3

Due: **10/1/2021 (Fri) 5:00PM @ Gradescope**

Issued: **9/22/2021 (Wed)**

Weight: **5% of total grade**

Note:

PS3-1 Image improvement via area-to-pixel filters

As discussed in class, the quality of an image can be improved by two types of area-to-pixel filters: (1) smoothing filters, and (2) sharpening filters. The former will reduce noise in an image, and the latter will make an image sharper.

In this problem, you are asked to find a smoothing filter, a sharpening filter, or a combination of them that improves the quality of each of the noisy and/or blurry images shown in Figure 1. Noisy/blurry image are shown on the left-hand side, and part of their ground truth images are shown on the right-hand side.

Your task is to find a good combination of smoothing and sharpening filters and the order of applying them to improve each image and make it as close as possible to the ground truth.

You do not need to write your own filters for this problem. Instead, use any OpenCV smoothing functions, including cv2.blur(), cv2.boxFilter(), cv2.GaussianBlur(), cv2.medianBlur(), cv2.bilateralFilter(). For sharpening, you may use cv2.filter2D() with a sharpening kernel that you define. You may also create an "unsharp masking" effect by combining a Gaussian-smoothed image and the original image by using cv2.addWeighted().

Submission

To prepare for the submission of your work on Gradescope, create:

(1) a folder called "ps3-1," that contains the following files:

- source code file(s)
- improved images created by your program:
  - pcb-improved.png
  - golf-improved.png
  - pots-improved.png
  - rainbow-improved.png
- "readme.txt" file that includes:
  - Operating system
  - IDE you used to write and run your code
  - The number of hours you spent to finish this problem

(2) a PDF file that contains the printouts and screenshots of all the files in the ps3-1 folder. (Include, if any, the mathematical derivation and/or description of your method in the PDF file. Handwritten notes should be scanned and included in the PDF file.)

Figure 1. Four images to be improved by smoothing and/or sharpening (Left: noisy and/or blurry images, Right: ground truth images)

PS3-2 Edge detection

In this problem, you will detect edges in an image by two methods:

(1) Write your own Sobel filtering code in Python to detect edges, and

(2) Use the Canny edge detector in OpenCV to detect edges.

Either way, take as input the color images, “cheerios,” “professor,” “gear,” and “circuit,” shown in Figure 2, and convert them to binary images showing edges in black on a white background.

The inner workings of the Canny edge detector are explained in the 9/22 lecture. They are also explained in the appendix placed at the end of this document.

Since the Canny results change significantly depending on specified parameters (threshold1, threshold2, apertureSize, and L2gradient), use some GUIs such as slider bars and radio buttons to allow a user to iteratively change and apply a different combination of parameters and see the result on the screen. Use your program to find the best combination of parameters for each of the four images.

For each image, compare and discuss the results of the Sobel filtering and the Canny edge detection.

Submission

To prepare for the submission of your work on Gradescope, create:

(1) a folder called “ps3-2,” that contains the following files:

- source code file(s)
- binary images created by each method:
    - cheerios-sobel.png, cheerios-canny.png
    - professor-sobel.png, professor-canny.png
    - gear-sobel.png, gear-canny.png
    - circuit-sobel.png, circuit-sobel.png
- "readme.txt" file that includes:
    - Operating system
    - IDE you used to write and run your code
    - The number of hours you spent to finish this problem

(2) a PDF file that contains the printouts and screenshots of all the files in the ps3-2 folder. Also include a side-by-side comparison showing images of both methods, and add a brief discussion. (Include, if any, the mathematical derivation and/or description of your method in the PDF file. Handwritten notes should be scanned and included in the PDF file.)

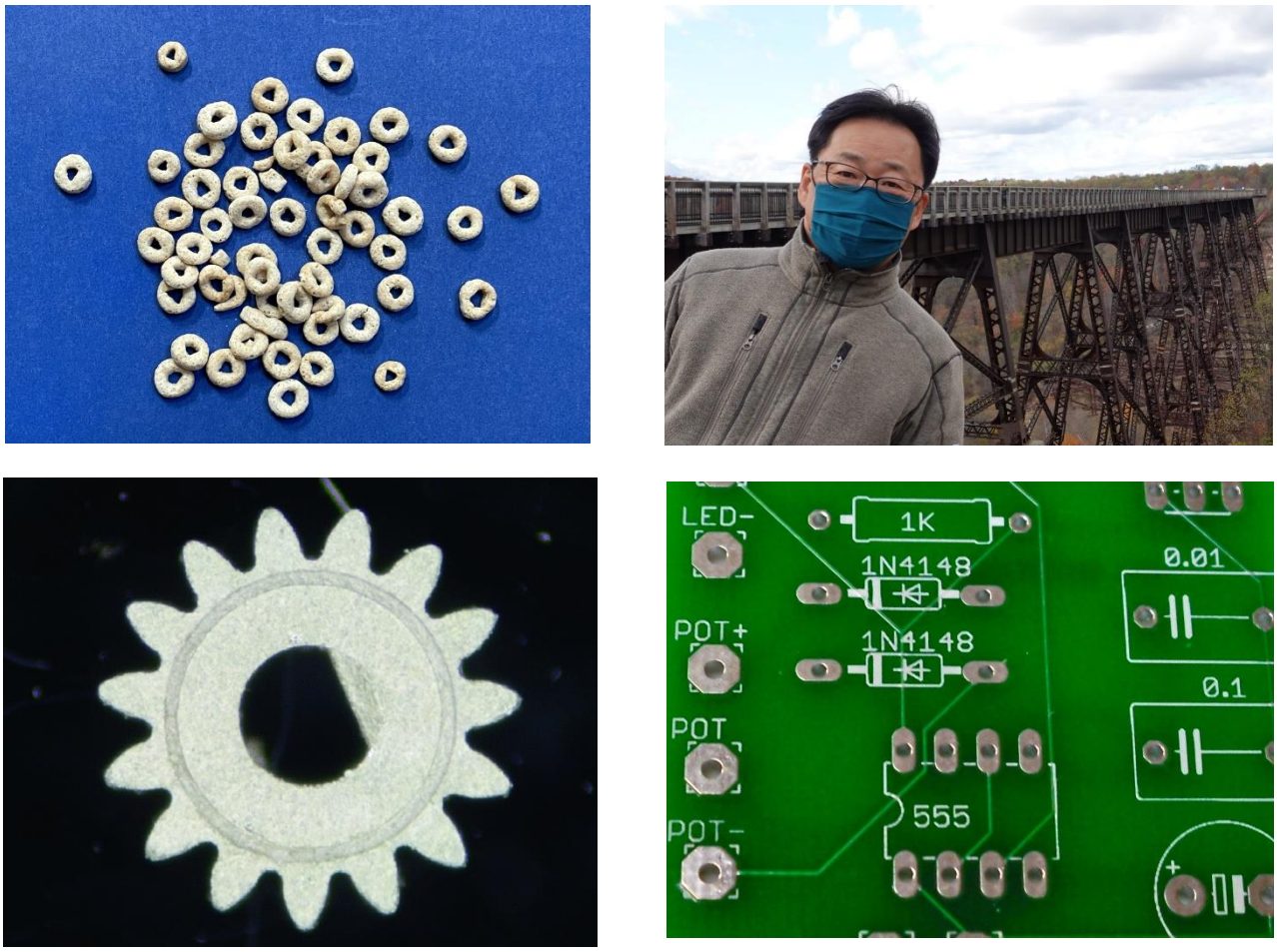


Figure 2. Take as input a color image and generate binary images showing edges

Submit your work on Gradescope

Submit two files on Gradescope – replace “andrewid” with your own Andrew ID:

(1) **andrewid-ps3-files.zip** – this ZIP file should contain the ps3-1 and ps3-2 folders and all the files requested.

(2) **andrewid-ps3-report.pdf** – this PDF file serves as the report of your work, and it should contain the printouts and screenshots of all the files in the “ps3-1” and “ps3-2” folders. (Include, if any, the mathematical derivation and/or description of your method in the PDF file. Handwritten notes should be scanned and included in the PDF file.)

Please organize pages with section titles and captions to make the report easy to read.

Appendix: Canny Edge Detection

(Source: https://docs.opencv.org/)

Canny Edge Detection is a popular edge detection algorithm. It was developed by John F. Canny in 1986. It is a multi-stage algorithm and we will go through each stages.

1. **Noise Reduction**

   Since edge detection is susceptible to noise in the image, first step is to remove the noise in the image with a 5x5 Gaussian filter. We have already seen this in previous chapters.

2. **Finding Intensity Gradient of the Image**

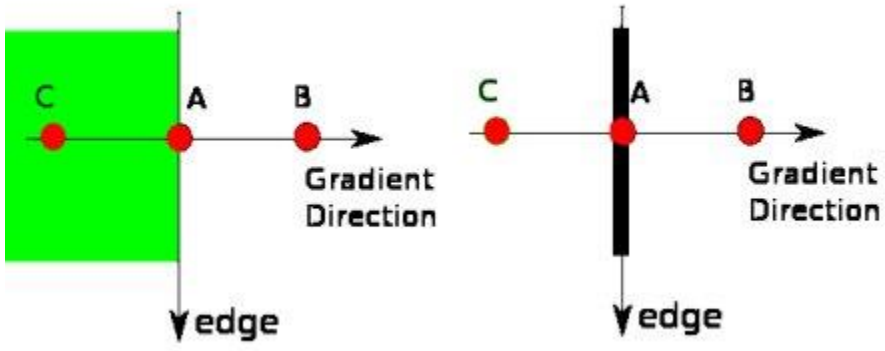
   Smoothened image is then filtered with a Sobel kernel in both horizontal and vertical direction to get first derivative in horizontal direction (G_x) and vertical direction (G_y). From these two images, we can find edge gradient and direction for each pixel as follows:

$$Edge\_Gradient\ (G) = \sqrt{G_x^2 + G_y^2}$$
$$Angle\ (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

   Gradient direction is always perpendicular to edges. It is rounded to one of four angles representing vertical, horizontal and two diagonal directions.

3. **Non-maximum Suppression**

   After getting gradient magnitude and direction, a full scan of image is done to remove any unwanted pixels which may not constitute the edge. For this, at every pixel, pixel is checked if it is a local maximum in its neighborhood in the direction of gradient. Check the image below:

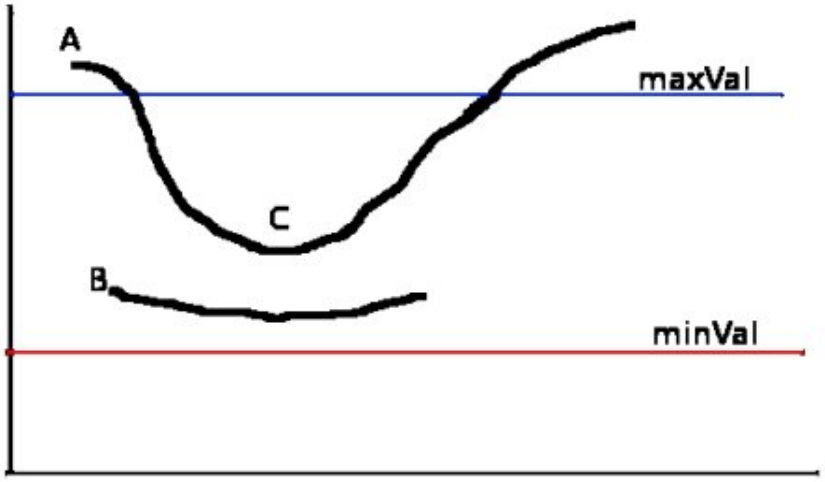


Point A is on the edge (in vertical direction). Gradient direction is normal to the edge. Point B and C are in gradient directions. So point A is checked with point B and C to see if it forms a local maximum. If so, it is considered for next stage, otherwise, it is suppressed (put to zero).

In short, the result you get is a binary image with "thin edges".

4. **Hysteresis Thresholding**

This stage decides which are all edges are really edges and which are not. For this, we need two threshold values, minVal and maxVal. Any edges with intensity gradient more than maxVal are sure to be edges and those below minVal are sure to be non-edges, so discarded. Those who lie between these two thresholds are classified edges or non-edges based on their connectivity. If they are connected to "sure-edge" pixels, they are considered to be part of edges. Otherwise, they are also discarded. See the image below:



The edge A is above the maxVal, so considered as "sure-edge". Although edge C is below maxVal, it is connected to edge A, so that also considered as valid edge and we get that full curve. But edge B, although it is above minVal and is in same region as that of edge C, it is not connected to any "sure-edge", so that is discarded. So it is very important that we have to select minVal and maxVal accordingly to get the correct result.

This stage also removes small pixels noises on the assumption that edges are long lines.

So what we finally get is strong edges in the image.

Canny Edge Detection in OpenCV

We use the function:

cv2.Canny(image, edges, threshold1, threshold2, apertureSize = 3, L2gradient = false)

Parameters

image	8-bit input image.
edges	output edge map; single channels 8-bit image, which has the same size as image.
threshold1	first threshold for the hysteresis procedure.
threshold2	second threshold for the hysteresis procedure.
apertureSize	aperture size for the Sobel operator.
L2gradient	specifies the equation for finding gradient magnitude. If it is True, it uses the equation mentioned above which is more accurate, otherwise it uses this function: $\text{Edge\_Gradient}(G)=\lvert G_x\rvert+\lvert G_y\rvert$.